Downloaded from orbit.dtu.dk on: May 05, 2023

DTU Library

# Fast Parametric Beamformer for Synthetic Aperture Imaging

**Nikolov, Svetoslav; Jensen, Jørgen Arendt; Tomov, Borislav Gueorguiev**

*Published in:*
IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control

*Link to article, DOI:*
10.1109/TUFFC.2008.860

*Publication date:*
2008

*Document Version*
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

*Citation (APA):*
Nikolov, S., Jensen, J. A., & Tomov, B. G. (2008). Fast Parametric Beamformer for Synthetic Aperture Imaging. *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, *55*(8), 1755-1767. https://doi.org/10.1109/TUFFC.2008.860

**General rights**
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

# Fast Parametric Beamformer for Synthetic Aperture Imaging

Svetoslav Ivanov Nikolov, Jørgen Arendt Jensen, *Senior Member, IEEE*, and Borislav Gueorguiev Tomov

***Abstract*—This paper describes the design and implementation of a real-time delay-and-sum synthetic aperture beamformer. The beamforming delays and apodization coefficients are described parametrically. The image is viewed as a set of independent lines that are defined in 3-D by their origin, direction, and inter-sample distance. The delay calculation is recursive and inspired by the coordinate rotation digital computer (CORDIC) algorithm. Only 3 parameters per channel and line are needed for their generation. The calculation of apodization coefficients is based on a piecewise linear approximation.**

**The implementation of the beamformer is optimized with respect to the architecture of a novel synthetic aperture real-time ultrasound scanner (SARUS), in which 4 channels are processed by the same set of field-programmable gate arrays (FPGA). In synthetic transmit aperture imaging, low-resolution images are formed after every emission. Summing all low-resolution images produces a perfectly focused high-resolution image. The design of the beamformer is modular, and a single beamformation unit can produce 4600 low-resolution images per second, each consisting of 32 lines and 1024 complex samples per line. In its present incarnation, 3 such modules fit in a single device. The summation of low-resolution images is performed internally in the FPGA to reduce the required bandwidth.**

**The delays are calculated with a precision of 1/16th of a sample, and the apodization coefficients with 7-bit precision. The accumulation of low-resolution images is performed with 24-bit precision. The level of the side- and grating lobes, introduced by the use of integer numbers in the calculations and truncation of intermediate results, is below −86 dB from the peak.**

## I. Introduction

Synthetic aperture (SA) focusing was conceived in the 1950s and has been implemented in radar systems using digital computers since the late 1970s [1]. Originally these systems used a single transmitting and receiving antenna, and the approach was known as monostatic SA imaging. The first attempts to build systems using ultrasound were made in the late 1970s and 1980s [2]–[7]. These systems were more or less a direct implementation of the monostatic SA imaging and had several limitations imposed by the technology. Poor signal-to-noise ratio and image quality prevented them from being used in medical scanners. From the beginning of the 1990s, more focus has turned toward the real-time implementation of synthetic aperture imaging [8]–[13] prompted by the advances in digital technology. During this period, several techniques to increase the signal-to-noise ratio have been introduced, such as the use of multiple elements in transmit [9], [10], [14]–[20], and new ways of calculating the propagation path have been defined [21]. The use of coded excitations has further increased the signal-to-noise ratio, image contrast, and penetration depth [22]–[26]. The real appeal of using SA methods for medical ultrasound is the use of synthetic transmit aperture (STA) focusing, where all transducer elements are used to receive at every transmission [19], [27]–[29]. The combination of all these approaches has made it possible to generate not only images at high frame rates, but also images that have superior contrast and resolution [30], [31].

In the last decade, effects of motion in SA imaging have been studied [32], and new methods for estimating the blood velocity were developed [19], [33]–[35]. In many cases, estimating the blood velocity using STA focusing performs better than traditional methods due to the uninterrupted flow of information for every point in the image. The methods are complicated to implement as they require beamforming along the flow direction, which may change from point to point in the image.

All these advances have spurred new interest in the real-time implementation of STA imaging [36]. The most critical problem is the beamformer and the very large number of points to generate.

This paper describes a parametric delay-and-sum beamformer capable of creating conventional 2-D ultrasound images, 3-D ultrasound volumes, and images acquired using synthetic aperture techniques. Among these applications, synthetic aperture imaging imposes the largest computational demands, which has influenced the architecture of the beamformer. The main challenge addressed in this work is the large number of image points beamformed per second—about $2 \times 10^9$ or 200 lines $\times$ 1000 samples/line $\times$ 5000 emissions/sec $\times$ 2 in-phase and quadrature phase (IQ) samples using signals from 256 channels acquired from up to 256 emissions. The delays and apodization coefficients used in the beamformation process are unique for every channel and transmission, and reading them from look-up tables exceeds 2 times the available bandwidth in the system. Using a flow-through architecture common for most scanners exceeds the available bandwidth 3 times. These 2 problems have been solved by suitable partitioning of processing and calculating the delay and apodization coefficients at run-time. The beamformer is implemented in an experimental scanner [37], [38], which will be used to test the clinical relevance of the developed STA methods.

Manuscript received December 13, 2007; accepted March 10, 2008. This work was supported by grant 26-04-0024 by the Danish Science Foundation and B-K Medical Aps, Herlev, Denmark.

The authors are with the Department of Electrical Engineering, Technical University of Denmark, Lyngby, Denmark (e-mail: sn@elektro.dtu.dk).

Digital Object Identifier 10.1109/TUFFC.2008.860

0885–3010/$25.00 © 2008 IEEE

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.

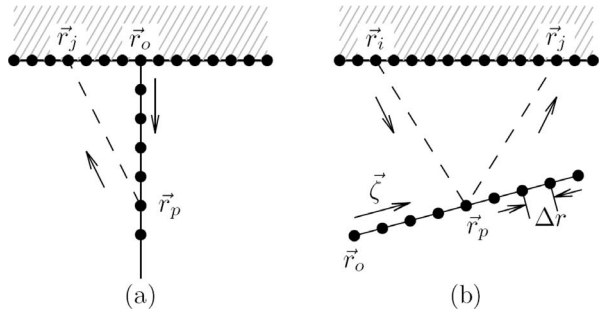


Fig. 1. Geometry used to calculate the delays: (a) conventional linear/phased array imaging, and (b) synthetic transmit aperture imaging. Solid line illustrates a beamformed line. Dashed line illustrates the propagation of the ultrasound wave.

The paper is organized as follows. Section II describes the algorithms used in the beamformer and its overall design. Section III gives implementation details in terms of data flow and required hardware resources. All calculations are done using integer numbers (fixed-point numbers) and the errors introduced are analyzed in Section IV. Beamforming results, shown in Section V, comply with the expected side-lobe levels predicted by the error analysis. Section VI concludes the paper.

## II. Beamformer Design

The beamformer consists of several beamforming units, each processing data from a limited number of channels and producing a part of the image. Fig. 1 gives an overview of a single beamforming unit. An image is specified as a set of lines to accommodate both SA and conventional imaging, and each unit can beamform 8 lines in parallel using data from 4 channels. It contains 5 delay calculation units—4 for the receive propagation time and 1 for the transmit propagation time. The unit contains also 5 units calculating the apodization coefficients. Both the apodization coefficients and the propagation times are calculated parametrically. Linear interpolation is used to generate samples with subsample delay. The sum of 4 receive channels represents a partial low-resolution image in the case of SA imaging. It is multiplied by the coefficient for transmit apodization before it summed with the rest of the low-resolution images. Several line-buffers accumulate the result until all transmissions have finished. The following sections describe each of these units.

### A. Line and Image Definition

The goal of this work has been to design and implement a beamformer that can handle arbitrary image and line geometry. It has been decided that images are defined as sets of lines and that each line consists of equispaced points as shown in Fig. 1. An image line is described by an origin $\vec{r}_o$, direction $\vec{\zeta}$, distance between successive points $\Delta r$ and the number of points in the line:

$$\begin{aligned}\vec{r}_p &= \vec{r}_o + p\vec{\zeta}\Delta r \\ &= \vec{r}_o + p\Delta\vec{r}.\end{aligned} \tag{1}$$

Another design objective is that the beamformer should handle both conventional beamforming, in which a focused transmit beam is created, and synthetic aperture imaging, in which spherical waves are transmitted and an image is created by coherent summation of data acquired over several transmissions. Delay-and-sum beamforming has the needed flexibility to accommodate both imaging situations. For phased and linear array images, only data from a single emission are beamformed to create a line or a set of lines in the case of a parallel beamformer:

$$s(\vec{r}_p) = \sum_{j=1}^{N} w_j(\vec{r}_p) g_j(t_{ToF}(\vec{r}_p, \vec{r}_o, \vec{r}_j)), \tag{2}$$

where $g_j(t)$ is the signal received by the $j$th element, $w_j$ is the dynamic apodization coefficient, and the time of flight is

$$\begin{aligned}t_{ToF}(\vec{r}_p, \vec{r}_o, \vec{r}_j) &= \underbrace{\frac{|\vec{r}_p - \vec{r}_o|}{c}}_{\text{transmit}} + \underbrace{\frac{|\vec{r}_p - \vec{r}_j|}{c}}_{\text{receive}} \\ &= \frac{l_{po}}{c} + \frac{l_{pj}}{c} \\ &= t_{po} + t_{pj}\end{aligned} \tag{3}$$

where $\vec{r}_p$, $\vec{r}_o$, and $\vec{r}_j$ are the coordinates of the beamformed point, the origin of transmission and the receive element, respectively. The geometry is illustrated in Fig. 1(a). The speed of sound is $c$ and the number of elements used in receive is $N$.

In synthetic aperture imaging, a full low-resolution image is formed after every transmission [19], and these low-resolution images must be added coherently to form the final high-resolution image. A double summation is thus needed to form a single point:

$$s(\vec{r}_p) = \sum_{i=1}^{M} \sum_{j=1}^{N} w_{ij}(\vec{r}_p) g_{ij}(t_{ToF}(\vec{r}_p, \vec{r}_i, \vec{r}_j)), \tag{4}$$

where $M$ is the number of transmissions, $i$ is the index of the transmitting element, and $w_{ij}$ is an apodization coefficient that depends both on the position of the transmit and receive elements. Notice that the propagation time is now calculated from $\vec{r}_i$, the position of the transmitting element, rather than $\vec{r}_o$, the origin of the line as it was done in (3).

The envelope of the beamformed signal is used both to form a B-mode image and to estimate the blood velocity. Thus, both the in-phase and quadrature-phase signals are needed in the beamformation process.

At every emission all points in the image are beamformed using all channel data, and a low-resolution image is formed. The resulting low-resolution image is accumulated in a buffer for the number of emissions $M$. Large

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.

buffers are needed for the storage of input data and intermediate results. This precludes the use look-up tables as done in [36]. Instead we have developed a recursive procedure for the calculation of time-of-flight.

### *B. Recursive Calculation of Time-of-Flight*

There exist many approaches to calculating the time of flight such as the ones described in [39], [40]. Most designs use either some form of symmetry or treat special cases. In this work, we want to calculate general propagation times in 3-D space. This enables beamforming along the direction of flow, which is used for vector flow estimation [35], [41].

The time of flight is made of 2 components $t_{ip}$ and $t_{jp}$, where $i$, $j$, and $p$ indicate indexes of transmit element, receive element, and a point along a line. Each unit calculating time of flight calculates the time of flight from a single element to a point along the line, and the index of the transmit or receive element will be skipped in the following. The calculation procedure is given by

$$t_p = \frac{l_p}{c} = \frac{|\vec{r}_o - \vec{r}_i + p\Delta\vec{r}|}{c} = \frac{|\vec{r}_{oi} + p\Delta\vec{r}|}{c}, \tag{5}$$

where $p$ is the index of the point along the image line ($0 \le p < P$) and $\vec{r}_{oi}$ is the position of the origin of the line expressed in a coordinate system, whose origin coincides with the center of the transducer element. The squared distance is given by

$$l_p^2 = x_p^2 + y_p^2 + z_p^2.$$

For each of the 3 coordinates, the squared distance can be written as

$$\begin{aligned} x_p^2 &= (x_{oi} + p\Delta x)^2 \\ &= x_{oi}^2 + 2px_{oi}\Delta x + (p\Delta x)^2. \end{aligned} \tag{6}$$

The value at the previous point $p-1$ is

$$x_{p-1}^2 = x_{oi}^2 + 2(p-1)x_{oi}\Delta x + ((p-1)\Delta x)^2. \tag{7}$$

Subtracting (7) from (6) gives

$$\begin{aligned} x_p^2 - x_{p-1}^2 &= 2x_{oi}\Delta x + (2p-1)\Delta x^2 \\ &= A_x + (2p-1)B_x \\ &= C_x + pD_x, \end{aligned} \tag{8}$$

where

$$A_x = 2x_{oi}\Delta x, \;\; B_x = \Delta x^2, \;\; C_x = A_x - B_x, \;\; D_x = 2B_x$$

are constants unique for a given combination of line and element and are passed as parameters to the delay calculation unit. The calculation of the difference between the squared propagation times expressed in number of samples, $d_p = f_s t_p$, is straightforward:

$$\Lambda_p = d_p^2 - d_{p-1}^2 = \frac{f_s^2}{c^2}\left(l_p^2 - l_{p-1}^2\right) = C + pD, \tag{9}$$

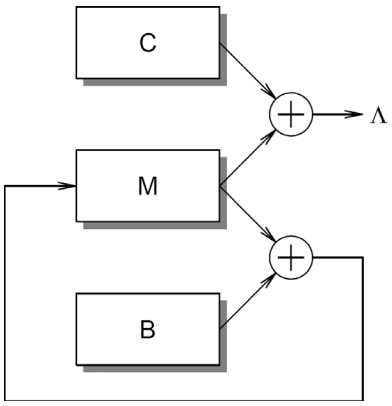


Fig. 2. Calculation of the difference between the square of the distances between 2 points.

where

$$C = A - B, \quad D = 2B \tag{10}$$

$$A = \frac{2f_s^2}{c^2}\left(x_{oi}\Delta x + y_{oi}\Delta y + z_{oi}\Delta z\right) \tag{11}$$

$$B = \frac{f_s^2}{c^2}\left(\Delta x^2 + \Delta y^2 + \Delta z^2\right), \tag{12}$$

and $f_s$ is the sampling frequency. The squared propagation time can be found by accumulating the following differences:

$$d_p^2 = d_{p-1}^2 + \Lambda_p. \tag{13}$$

The circuit that finds $\Lambda_p$ is shown in Fig. 2. It consists of 3 registers. The product $pD$ is found by adding $D$ to the register $M$ at every step.

What is needed is the propagation time $d_p = \sqrt{d_p^2}$. Several methods exist to find the square root of a number such as Newton's method [42], a coordinate rotation digital computer (CORDIC) processor [43], [44], and recursive iterative calculation (RISQRT) [45]. The RISQRT method was previously found to occupy the least amount of resources [45] and is the one that has been chosen for implementation. The calculation procedure is described in the following paragraphs.

It can be shown that $|d_p - d_{p-1}| \le \Delta d$, where $\Delta d = f_s|\Delta\vec{r}|/c$. Equality occurs only when the propagation path is along the image line. The idea of the algorithm is to find an approximation $\hat{d}_p$ of $d_p$ in several iterations. The current value of the approximated value of $d_p$ at iteration $m$ will be denoted by $\hat{d}_p[m]$. The algorithm makes sure that the first approximation $\hat{d}_p[0]$ is less than the desired $d_p$. At every iteration $m$, the algorithm calculates a new candidate for $\hat{d}_p[m]$ $\delta[m]$. If $\delta[m]^2 < d_p^2$, then the result of the $m$th iteration is $\hat{d}_p[m] = \delta[m]$, otherwise it remains unchanged $\hat{d}_p[m] = \hat{d}_p[m-1]$. The procedure is summarized as follows:

```
1: if d̂²_{p-1} > d²_p then
2:     d̂_p[0] := d_{p-1} − Δd
```

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.

```
3: else
4:    d̂_p[0] := d_{p-1}
5: N_b := ⌈log2 Δr⌉
6: for m := 1 to N_b do
7:    begin
8:    δ[m] := d̂_p[m-1] + Δr/2^m
9:    if (δ[m])^2 < d_p^2 then
10:      d̂_p[m] := δ[m]
11:   else
12:      d̂_p[m] := d̂[m-1]
13: end
14: d_p := d̂_p[N_b]
```

The outlined procedure needs multiplications and comparisons. These can be avoided by keeping track of both $\hat{d}_p[m]$ and the difference

$$\hat{\Delta}_p[m] = \hat{d}_p^2[m] - d_p^2. \tag{14}$$

The condition that $d_p[m]^2 < d_p^2$ is equivalent to the condition that $\hat{\Delta}_p[m] < 0$. Using binary numbers, this is equal to examining the sign bit of the difference. The starting value of $\hat{\Delta}_p[m]$, when $\hat{d}_p[0] = d_{p-1}$, will be $\hat{\Delta}_p[0] = -\Lambda_p$. For the value of $d_{p-1}$, however, we use the result from the last iteration $\hat{d}_{p-1}[N_b]$, which differs from the real value of $d_{p-1}$. To avoid the accumulation of errors, the initial value of $\hat{\Delta}_p[0]$ is

$$\hat{\Delta}_p[0] = -\Lambda_p + \hat{\Delta}_{p-1}[N_b]. \tag{15}$$

The multiplications can be avoided by making sure that only multiplications by $2^x$ are used. The initial value $\Delta d = 2^{N_b}$, where $N_b$ represents the number of bits needed to encode $\Delta d$. At every iteration, we form $\delta[m] = d_p[m-1] + \varepsilon[m]$, where $\varepsilon[m] = 2^{N_b - m}$. One can calculate $\hat{\Delta}_p[m]$ from $\hat{\Delta}_p[m-1]$ as follows:

$$\begin{aligned}
\hat{\Delta}_p[m] &= \hat{d}_p^2[m] - d_p^2 \\
&= \left(\hat{d}_p[m-1] + \varepsilon[m]\right)^2 - d_p^2 \\
&= \hat{d}_p[m-1]^2 + 2d_p[m-1]\varepsilon[m] + \varepsilon^2[m] - l_p^2 \\
&= \hat{\Delta}_p[m-1] + 2d_p[m-1]\varepsilon[m] + \varepsilon^2[m] \\
&= \hat{\Delta}_p[m-1] + 2^{N_b - m + 1} d_p[m-1] + 2^{2(N_b - m)}.
\end{aligned} \tag{16}$$

The procedure to find the propagation times $d_p$ expressed in number of samples thus becomes

```
1: {Input data: d̂_{p-1} - estimate of d_{p-1}}
2: {Input data: Δ̂_{p-1} - residual error d̂_{p-1}^2 - d_{p-1}^2}
3: {Input data: Λ̂_{p-1} - difference d_p^2 - d_{p-1}^2}
4:
5: {Output data: d̂_p - estimate of d_p}
6: {Output data: Δ̂_p - residual error d̂_p^2 - d_p^2}
7: N_b := ⌈log2 Δd⌉ is the number of bits needed for Δd}
8:
9: {Ensure that d̂_p[0] < d_p}
10: if (Λ_p < 0) then
11:    begin
12:    d̂_p[0] := d̂_{p-1}
13:    Δ̂_p[0] := Λ_p + Δ_{p-1}
14:    end
15: else
16:    begin
17:    d̂_p[0] := d_{p-1} - 2^{N_b}
18:    Δ̂_p[0] := Δ_{p-1} - 2^{N_b+1} d̂_{p-1} + 2^{2N_b}
19:    end
20:
21: for m: = 1 to N_b do
22:    begin
23:    k := N_b - m
24:    δ[m] := d̂_p[m-1] + Δr/2^k
25:    D[m] = Δ̂_p[m-1] + 2^{k+1} d_p[m-1] + 2^{2k}
26:    if (D[m] < 0) then
27:    begin
28:    d̂_p[m] := δ[m]
29:    Δ̂_p[m] := D[m]
30:    end
31: else
32:    begin
33:    d̂_p[m] := d̂[m-1]
34:    Δ̂_p[m] := Δ̂[m-1]
35:    end
36: end
37:
38: { Output result }
39: d̂_p := d̂_p[N_b]
40: Δ_p := Δ̂_p[N_b]
```

Two implementations of this circuit are possible—iterative and pipelined. The former outputs a new result at every $N_b$ clock cycles, and the latter outputs a result at every clock cycle. The calculation is recursive, and the pipelined circuit needs to calculate delays for $N_b$ lines interleaved as illustrated in Fig. 3, where 8 lines are being beamformed simultaneously. The processing that is performed in a single stage of the pipeline is shown in Fig. 4. Initially the new candidates for $\hat{\Delta}_p[m]$ and $\hat{d}_p[m]$, $\mathcal{D}_m$ and $\delta[m]$, are calculated. Based on the sign bit of $\mathcal{D}[m]$, the registers with $\hat{\Delta}$ and $\hat{d}$ are updated or not. Because the number of bits is chosen at design time, the multiplication by $2^k$ is reduced to appropriate wiring.

### C. Apodization

Apodization curves must be calculated separately for every combination of line and channel. The apodization curves are not fixed, and it has been found that piecewise linear interpolation describes the curves with sufficient precision. Each segment is described by a start value, increment, and number of samples. The start value prevents the accumulation of errors from the use of integer numbers. Furthermore, discontinuities are handled in an easy manner.

### D. Interpolation

To achieve higher precision, subsample delays must be used. This is achieved using interpolation [46], [47]. Linear interpolation is used to generate samples for time instances that are not integer multiples of the sampling period. It gives better results than nearest neighbor interpolation [36] and is still relatively simple to implement.

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.

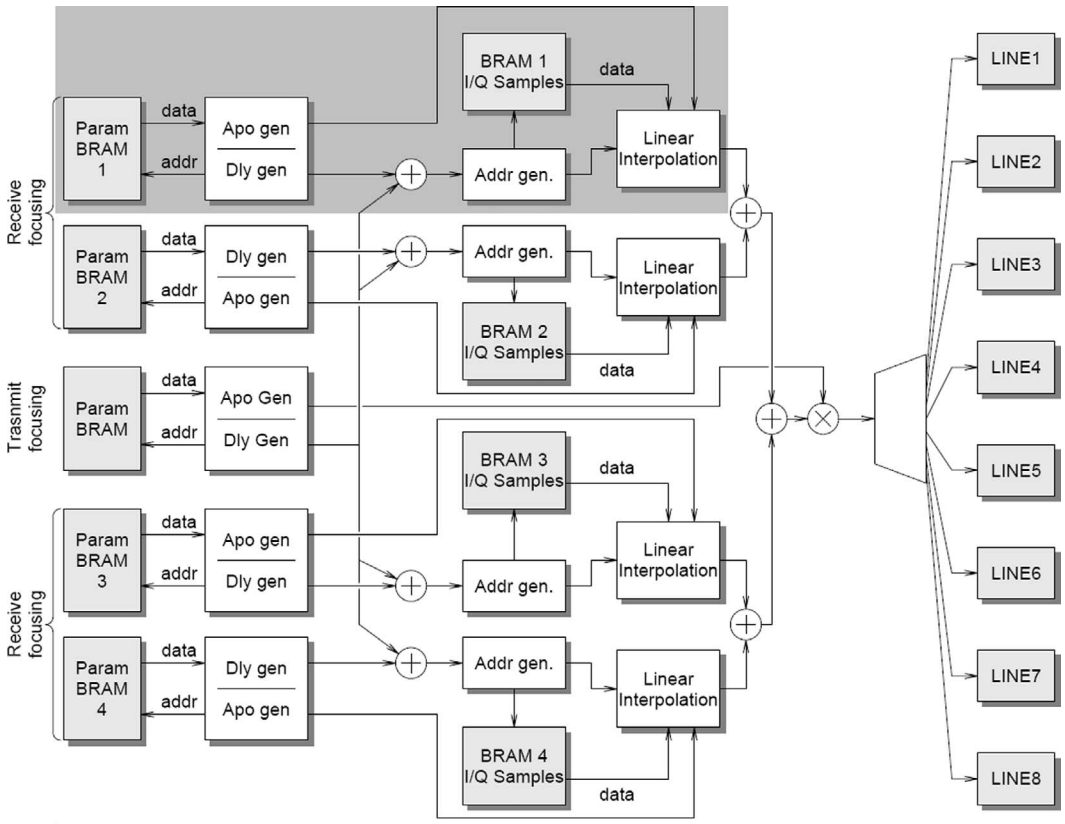


Fig. 3. Overview of the beamformer's architecture. The beamformer is divided into blocks. Each block creates a part of a high-resolution SA image using data from 4 receive channels. Data are sent further in the system after all transmissions have been carried out.

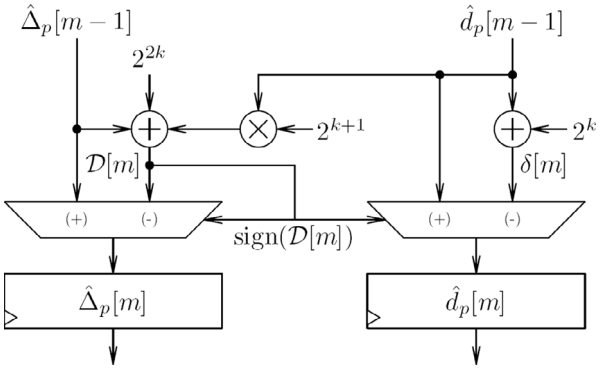


Fig. 4. A single stage of the RISQRT pipeline.

The propagation time $d_p$ consists of an integer part and a fraction:

$$d_p = n + \alpha. \tag{17}$$

The range of the integer part $n$ is determined by the maximum number of samples stored by the system. The output from the beamforming is

$$\begin{aligned} s[p] &= \alpha g[n+1] + (1-\alpha) g[n] \\ &= g[n] + \alpha(g[n+1] - g[n]), \end{aligned} \tag{18}$$

where $g[n]$ is the sampled signal of the channel that is processed, $p$ is the index of the beamformed sample along the image line. For every output sample, 2 input samples must be read from the buffer memory. The interpolation unit must therefore work at twice the clock frequency as the rest of the circuit. Fig. 5 shows a block diagram of the interpolation circuit. Two registers are used to store the address $n$ and the interpolation coefficient $\alpha$. The sample at address $n$ is fetched in register $R1$ during the first half-period of the clock cycle. During this period, the address $n+1$ is formed, and $g[n+1]$ is read in $R2$. Both $R1$ and $R2$ are clocked at the double clock frequency. The register for the result is clocked at the working clock frequency. This allows for an uninterrupted output of samples $s[p]$, which are produced at every clock cycle of the fundamental clock. The total delay in the circuit is 1 clock cycle.

## III. Implementation

This section gives the implementation details. The goal has been to implement a versatile beamformer capable of beamforming 40 high-resolution images per second consisting of up to 192 lines and 1024 samples/line. Each high-resolution image is made by adding together the beamformed low-resolution images from up to 128 emissions. The analog-to-digital converter operates at a sampling frequency of 75 MHz, and the basic clock frequency has been chosen to be 150 MHz. The processing has been distributed over 64 boards. Data from 256 channels are

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.

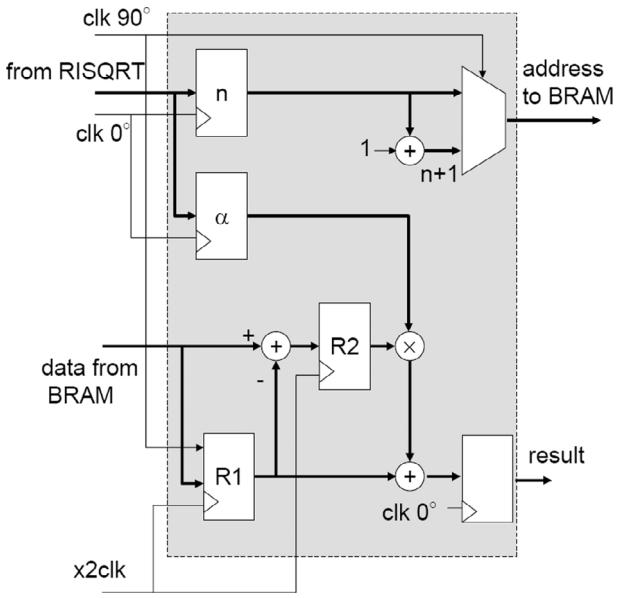


Fig. 5. Interpolation circuit. Linear interpolation requires that 2 input samples are read for each output sample. The circuit is clocked at twice the clock rate for the rest of the design.

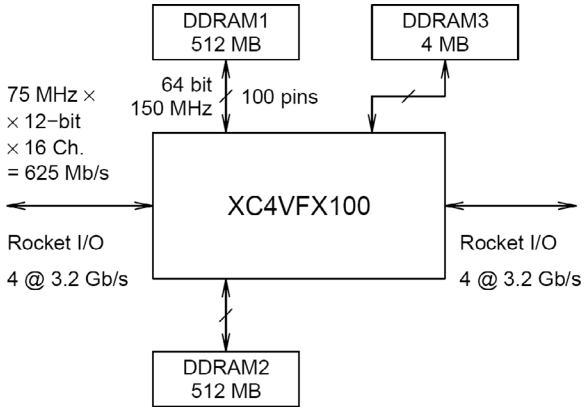


Fig. 6. The beamformer is comprised of several identical building blocks. Each of them features an FPGA, 2 SDRAM blocks of 512 MB for data processing and a single 4 MB SDRAM block for parameters. The read/write operations are performed at a double data rate.

used to create synthetic-aperture images in real time. The total number of beamformed samples per second is 881 million. Each beamformation unit can process 150 million IQ samples hence, 6 beamformation units per board are needed. As it will be shown below, 3 beamformation units can be fitted in a Virtex-4 (Xilinx, Inc., San Jose, CA) field-programmable gate array (FPGA), used in the system. The beamformation of high-resolution images from 4 receive channels is therefore distributed over 2 FPGAs.

Fig. 6 shows one of the FPGAs. The device chosen for the implementation is XC4VFX100 (Xilinx), and its logic resources, which are essential to the design of the beamformer, are shown in Table I. The serial transceivers are configured to operate at 3.2 Gb/sec. The data communication between every 2 FPGAs is implemented with 4 serial links, giving a bandwidth of 12.8 Gb/s. Separate

TABLE I
RESOURCES ESSENTIAL TO THE DESIGN AND AVAILABLE IN THE CHOSEN FIELD-PROGRAMMABLE GATE ARRAY XC4VFX100 [48].

| Feature | Amount |
|---|---|
| Logic Cells | 94896 |
| Block RAM/FIFO (18 kbits each) | 376 |
| XtremeDSP$^{TM}$ Slices | 160 |
| RocketIO Serial Transceivers | 20 |

links (not shown) are used for parameter distribution and setup. Two 512 MB synchronous dynamic random access memory (SDRAM) buffers are connected to the FPGA for the purpose of storing data. The read/write operations are performed at double data rate, making it possible to achieve a peak bandwidth of $2 \times 150 \times 10^6 \times 8 = 2.4 \times 10^9$ bytes/second. A third SDRAM holds the values of the imaging parameters needed to calculate the time of flight and the apodization coefficients.

The buffers for parameters, IQ samples and beamformed lines are held internally in the FPGA using the block RAMs. The multiplications and some of the additions are implemented using the dedicated digital signal processing (DSP) blocks, which can perform a multiply-and-accumulate operation in a single clock cycle [49].

The limitations imposed by the resources are detailed in the following paragraphs.

### *A. I/O Bandwidth*

The data, sampled at 75 MHz, are consequently downsampled so that the channels' data do not exceed 4096 samples [37]. The samples are sampled at 12 bits/sample. Then they are matched-filtered, and the resulting size is 16 bit. Both IQ samples are created in the filtration process. The matched filtered samples are sent for further processing to another FPGA. The bandwidth $B_{\text{in}}$ of this communication link sets a limit on the highest achievable pulse repetition frequency $f_{\text{prf}}$; $f_{\text{prf}}$ is proportional to the input bandwidth of the FPGA[1], $B_{\text{in}}$, and is inversely proportional to the number of channels $N_{\text{chnl}}$, the number of input samples $N_s$, and the precision $N_{b_{\text{in}}}$:

$$\begin{aligned} f_{\text{prf}_{\max}} &= \frac{B_{\text{in}}}{2N_s N_{b_{\text{in}}} N_{\text{chnl}}} \\ &= \frac{4 \cdot 3.2 \cdot 2^{30}}{2 \cdot 2^{12} \cdot 16} \\ &= 26 \text{ kHz}. \end{aligned} \tag{19}$$

A typical imaging situation will have a $f_{\text{prf}}$ between 3000 Hz and 10 000 Hz, which leaves a comfortable margin.

The target is to create up to 40 high-resolution frames per second to capture fast movements. One FPGA houses 3 beam-former blocks, as shown in Fig. 3, and processes half image. The required output bandwidth is thus

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.

$$B_{\text{out}} = 2 \cdot \frac{N_l}{2} \cdot N_p \cdot N_{b_{\text{out}}} \cdot f_r = 2 \cdot 96 \cdot 1024 \cdot 24 \cdot 40 = 180 \text{ Mb/s} \quad (20)$$

where $N_l$ is the number of beamformed lines, $N_p$ is the number of samples per line, $N_b$ is the number of bits per sample, and $f_r$ is the frame rate. It can be seen that the output bandwidth is well below the peak bandwidth of 12.8 Gb/s. Notice that the samples are represented by 24 bits to decrease the effects of truncation in the calculations (see also Section IV-C).

### *B. Block RAM*

A full low-resolution image is created for every emission. Then the low-resolution images are summed together to create a high-resolution image. Typically between 64 and 128 emissions are used to create a high-quality B-mode image. Two approaches are possible: 1) create low-resolution images in one FPGA and then send them for summation in another, and 2) accumulate them inside the FPGA. In this work, the latter approach has been adopted because of the high bandwidth the former scheme requires. All buffers shown in Fig. 3 are implemented using the dedicated block RAMs. They are synchronous and dual-ported and can be configured to address 1 bit, 4 bit, 8 bit, 16 bit, and 32 bit. They feature additional bits, so that one can use 36 bits per address.

To simplify the synchronization, double buffering (sometimes also called ping-pong buffering) is used to load the data. Buffers are needed to hold the parameters for the delay and apodization generation, the input IQ samples, and the output beamformed lines.

*1. Generation of Delays and Apodization Coefficients:* To calculate the delays, 5 parameters are needed per line per channel—the 2 constants $B$ and $C$, see (12) and (10); the propagation time to the first sample in the line $d_0$; and the number of samples $N_p$. The propagation times $d_p$ are calculated with 16-bit precision. The differences $\Lambda_p$ are calculated with 32-bit precision, which is sufficient [45]. The beamformer processes 8 lines in parallel. The required storage for the delays for 8 lines for one channel is 25 32-bit values: $1 \times N_p + 8 \times B + 8 \times C + 8 \times d_0$. A beamformer must process a total of 32 lines, requiring a total of 100 32-bit values.

The apodization is calculated using piecewise linear approximation. The number of line segments is limited to 8. For each line-segment, 36 bits are required. The slope and initial value are represented with 14-bit internal precision and the number of samples in a segment with 8 bits. The total number of 36-bit words is 32 lines $\times$ 8 segments = 256 36-bit words. The block RAMs can be configured as 512 addresses $\times$ 36-bit words. Hence, one block RAM is sufficient to describe all lines for a single channel. Every acquisition is performed using a different transmit element, and new parameters must be downloaded at every transmission using double buffering. A single beamformer block like the one in Fig. 1 requires 6 block RAMs.

*2. Input Samples:* Input samples must be stored while a full image is being beamformed. Inside the FPGA, the samples from 4 channels for 1 emission must be stored. It has been decided that each beamformation block operates independently and will have its own copy of the input samples. Up to 4096 IQ 16-bit samples are stored for each channel. Double buffering is used for communications. The number of RAM blocks per channel thus is 2 buffers $\times$ 2 bytes/sample $\times$ 4096 samples/2048 bytes/RAM block = 16 blocks of RAM. A single beamformer handles 4 channels and thus needs 64 RAM blocks.

*3. Output Samples/Line Buffers:* Output samples are stored in 24 bits to minimize the effects of rounding in the multiplication. A beamforming block has to process 32 lines, which is 1/6th of all the lines in an image. Storing all lines internally in the FPGA leaves room only for 2 beamforming blocks. Thus it has been decided that a beamformer block will store internally only 8 lines. Once they are beamformed, they are sent further while the beamforming block scans the next 32 lines. To beamform 8 lines fully, the information from all transmissions is needed. Because only the data from a single transmission is kept in the FPGA, the sampled data are first saved in one of the external DDR RAMs, while the data from the other DDR RAM are being processed. The amount of data needed for 1 image is 128 transmissions $\times$ 2(IQ) $\times$ 2 bytes/sample $\times$ 4096 samples/channel $\times$ 4 channels = 8 MB. The required bandwidth is 8 MB $\times$ 40 frames/s = 320 MB/s. The bandwidth is 2.4 GB/s (see Fig. 5), which makes it possible to read up to 8 times the data necessary to beamform an image or a part of it. The beamformer blocks create only 1/4th of what is required at a time, thus the sample data must be read 4 times, thus using 50% of the available bandwidth.

The memory buffers needed to store 8 lines is 2 buffers $\times$ 8 lines $\times$ 1024 samples/line $\times$ 3 bytes/sample $\times$ 2(IQ)/2048 bytes/RAM block = 48 blocks of RAM.

So one beamformation block uses a total of 118 RAM blocks—6 for parameters, 64 for input samples, and 48 for output samples. The total number of RAM blocks is 376 (see Table I) allowing for 3 beamformer blocks to fit in the FPGA.

### *C. DSP Slices*

The DSP slices are needed because of the built-in $18-\text{bit} \times 18-\text{bit}$ multipliers. These multipliers can operate at frequencies of up to 450 MHz and can perform a multiply-and-accumulate operation at every clock cycle. The data from a single channel is first interpolated as shown in Fig. 5 using one multiplier. Then the 8-bit receive apodization is multiplied onto the result. Finally, the summed data are multiplied by the transmit apodization before summation. Thus, a beamformer block requires

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.

TABLE II
TRANSDUCER PARAMETERS USED IN THE SIMULATIONS.

| Parameter | Value |
|---|---|
| Number of elements | 256 |
| Element width | 246 $\mu$m |
| Kerf | 62 $\mu$m |
| Pitch (kerf+width) | 308 $\mu$m |
| Center freq. $f_0$ | 5 MHz |
| Sampling freq. $f_s$ | 20 MHz |
| Speed of sound $c$ | 1540 m/s |
| Wavelength $\lambda$ | 308 $\mu$m |

10 Extreme DSP slices for the I channel and 10 Extreme DSP slices for the Q channel. The total number of Extreme DSP slices for 3 beamforming blocks is thus 60. The total number of available Extreme DSP slices is 160 as given in Table I.

### D. Logic Cells

Using the synthesis report generated by the Xilinx ISE tool, a beamformation block requires about 3400 to 3500 out of 42 176 available slices. Three beamformation blocks thus occupy about 25% of the available logic resources. The maximum clock frequency for a device speed grade 11 is 167.8 MHz in our implementation and is limited by the RISQRT pipeline logic.

## IV. Error Analysis

There are 2 errors introduced in the beamformation process—errors from the calculated time of flight and the calculated apodization coefficients.

### A. Errors in the Time of Flight

The results in this section are based on calculations using the parameters listed in Table II. To illustrate the errors, the time of flight to the points of a line with origin $\vec{r}_o = (0, 0, 0.003)^T$ m and direction $\vec{\zeta} = (0, 0, 1)^T$ has been used in this section. The number of points in the line was set to 1024 and the end depth to 0.139 m. Fig. 7 shows an example of how the error for elements with coordinates $\vec{r}_5 = (-0.038, 0, 0)^T$ m looks as a function of distance along the line. The error is characterized by a dc-offset that changes as a function of depth and a random ac-component. The middle plot shows how the dc-offset (bias) changes. It is due to the error accumulated recursively in the calculation of the squared distance; see (9) and (13). This bias varies from channel to channel; it can start from a negative value and move toward a positive value, or it can start from a positive value and move toward a negative value, or it can be 0. Subtracting this bias from the total error gives the error due to the RISQRT algorithm, which is shown in the bottom plot of Fig. 7. The time-of-flight $\hat{d}_p$ is calculated in samples using fixed-point numbers with 12 bit for integer part and 4 bit for the fractional part. The error due to RISQRT varies in the interval $[0 : 2^{-4}]$ samples.

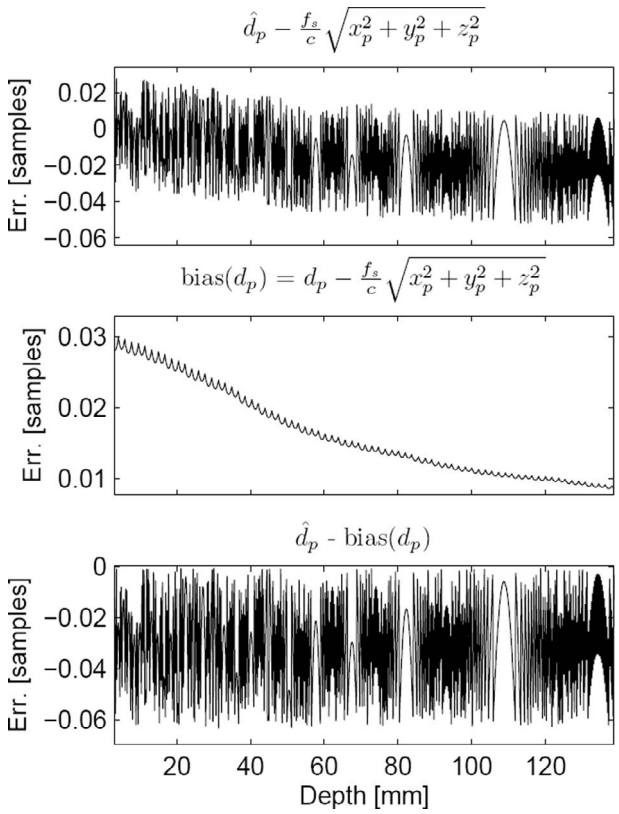


Fig. 7. Example of the error in the time-of-flight calculation for element with coordinates $\vec{r}_5 = (-0.038, 0, 0)^T$ m. Top plot shows the difference between the found $\hat{d}_p$ and the exact time-of-flight calculated using floating-point numbers with double precision. Middle plot shows the "bias" introduced by the use of integer numbers in the recursive calculation of the squared propagation time. Bottom plot shows the error that is introduced by the calculation of the square root.

Previously, Holm and Kristoffersen have given an analysis of the effect of quantizing the delays to the side lobes in a focused system [50]. In their work, they distinguish between random and periodic errors. Periodic errors give a rise to undesired grating lobes. To characterize the nature of the error, we look at the power density spectrum of the error, which is shown in Fig. 8. The figure shows the power-density spectrum of the error for a single point in a line across all channels. The spectrum has been averaged over all points in a line. The top graph shows the error caused by quantizing the exact time of flight with 4-bit precision for the fractional part. The error is white with uniform distribution. The expected level is $10\log((2^{-4})^2/12) = -34.88$ dB. The total error is not white, but the lack of dominant periodic components prevents the rise of grating lobes. The distribution of the error (not shown) can be described by a Gaussian function. The error introduced by the RISQRT algorithm is white in nature, and its power is the same as the power of a quantization error.

The delay calculation circuit has been tested for several imaging geometries—linear and phased array imaging and

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.

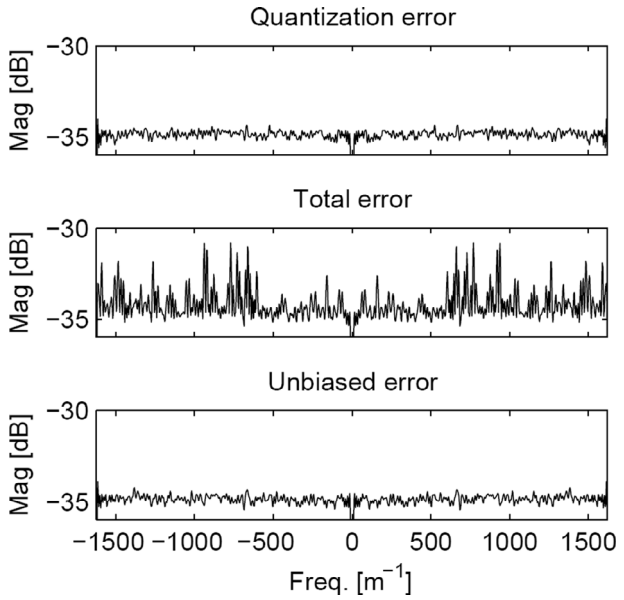


Fig. 8. Power density spectra of the error. Top: power density spectrum of the error from quantization of the *exact* time of flight. Middle: power density spectrum of the total error. Bottom: power density spectrum of the error with the bias removed.

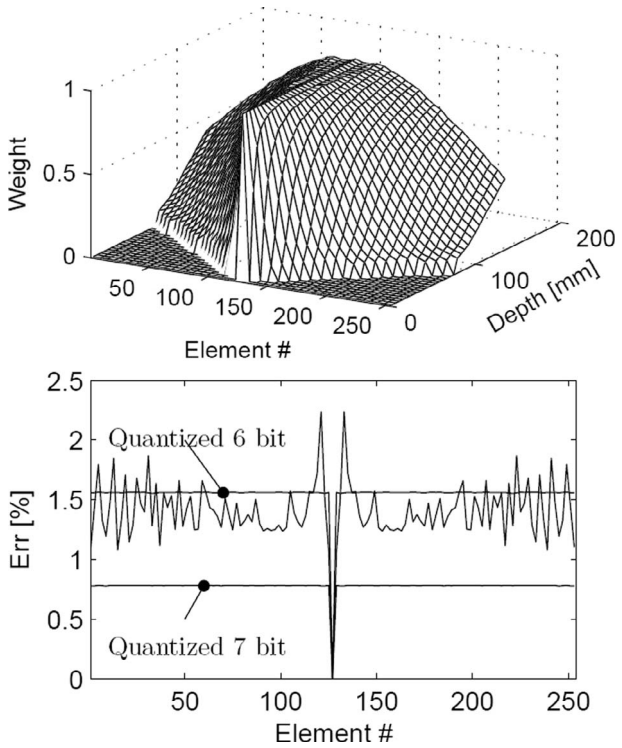


Fig. 9. Example of the approximation of dynamic apodization based on a Hamming window. The top plot shows the calculated weight coefficients as a function of depth and element number. The bottom plot gives the maximum absolute error. For comparison, the maximum absolute errors when quantizing with 7 and 6 bits are shown too.

beamforming for directional flow estimation. These tests are illustrated in [45]. The mean absolute error for all these cases is half of the least significant bit ($2^{-5}$). The sampling frequency is set to $f_s \geq 4f_0$, which means that the mean absolute error is on the order of $\lambda/128$. The reason is that for the same channel, the bias—as seen in the top plot in Fig. 7—changes from line to line and has a mean of zero. Thus, the only error that is always present is the error from the RISQRT circuit, which in magnitude and probability distribution is equal to the error introduced by quantization of the exact time of flight.

### *B. Errors in Calculation of Apodization*

We have chosen to use 7 significant bits to encode the apodization (plus a sign bit). The slope is encoded with 14 bits to minimize the effect of accumulated error. This error manifests itself as a discontinuity at the point when a new segment starts. Fig. 9 shows an example of dynamic apodization based on expanding Hamming window. The parameters of the transducer are the same as in the previous section (see Table II). The top plot shows a view of the approximated apodization as a function of depth and element number. The bottom plot shows the maximum absolute error of the calculation. For comparison, the maximum errors when the apodization is quantized with 6 and 7 significant bits are shown too. The mean absolute error tends toward the mean absolute error of 7-bit quantization. For a given point, the calculated apodization curve is "smooth." The average increase in the side-lobe level of the magnitude spectrum of the apodization function is less than 1 to 2 dB.

### *C. Errors from Truncation of Intermediate Calculations*

The data format for the sampled data are 16 bit, but data are sampled with 12-bit precision. The linear interpolation uses the built-in $18 \times 18$ multipliers and adds 4-bit precision from the interpolation. The 16-bit result is multiplied by the 7-bit apodization coefficient giving a 23-bit result without any rounding employed. The data from the 4 channels are added together without any loss of precision adding extra 3 bits. The 17 most significant bits of the sum are then multiplied by the 7-bit transmit appodization. The system is dimensioned so that 128 transmissions can be added together without loss of precision. Again, only the 17 most significant bits of the weighted sum are accumulated in the line buffers, which are 24 bits wide. Thus, there are 2 sources of noise from truncation—one before the multiplication with transmit apodization and one before the accumulation to create the final high-resolution image. Assuming that the apodization coefficients are 1, then the total power of the error is

$$P_{\text{accum}} = \sum_{i=1}^{M} 2P_{17\text{-bit}}, \tag{21}$$

where $M$ is the number of transmissions, and $P_{17\text{-bit}}$ is the noise of rounding the result at 17 bits. It is also assumed that the error can be modeled as white noise with uniform distribution. As mentioned before, 4 channels are

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.

processed on a single board. To create the final image, the results from each board are sent to be added to the sum from the other board. The system is dimensioned for up to 256 channels, distributed over 64 boards. The sum is 16 bit. The summation over 64 boards adds 6 bits to the result. The result from accumulating 4 channels must be downshifted, so that only 10 bits remain to prevent overflow. The noise from this cascade operation, when the resulting sum is 16 bit, is

$$P_{\text{cascade}} = \sum_{k=1}^{N/4} P_{\text{10-bit}}, \tag{22}$$

where $P_{\text{10-bit}}$ is the noise introduced by rounding the result at 10 bits. The total noise is the sum of the 2 components. For a system with $N = 256$ receive elements (64 boards) and $M = 128$ transmissions, the total noise is

$$\begin{aligned} P_{\text{16-bitsum}} &= 128 \cdot 2 \cdot \frac{\left(2^{-17}\right)^2}{12} + 64 \cdot \frac{\left(2^{-10}\right)^2}{12} \\ &\approx -53 \text{ dB}. \end{aligned} \tag{23}$$

This implies that the side-lobe level will be dominated by the noise introduced by the rounding operations.

Another implementation is to use a 24-bit accumulation. In this case, the result from summation must be rounded at 17-bit level. The power of the noise, thus becomes

$$\begin{aligned} P_{\text{24-bitsum}} &= 128 \cdot 2 \cdot \frac{\left(2^{-17}\right)^2}{12} + 64 \cdot \frac{\left(2^{-17}\right)^2}{12} \\ &\approx -86 \text{ dB}. \end{aligned} \tag{24}$$

This level is lower than the typical display range of 60 dB as shown in Fig. 10.

## V. Imaging Results and Discussion

Fig. 10 shows a B-mode image of a sector scan formed by a 128-element transducer using synthetic aperture focusing with 128 transmissions. The signals for the beamformer have been created by a simulation using Field II [51], [52]. The transducer parameters are the same as in Table II, except for the number of elements, which is 128, and the pitch, which is $\lambda/2$.

The goal of the simulation is to reveal the influence of the different imaging parameters. A reference image has been created in which the input samples are represented with floating point numbers with double precision. The result from the implemented beamformer is shown in the bottom image of Fig. 10. The input signal has been quantized with 12 bits (range: $-2^{11}, 2^{11} - 1$). The precision of the delays and apodization is as described in the previous sections. The delays were calculated with 12 bits for integer part and 4 bits for the fractional part. The apodization was calculated with 7-bit signed numbers. The samples have been right-shifted by 11 bits before accumulation in the line buffers.

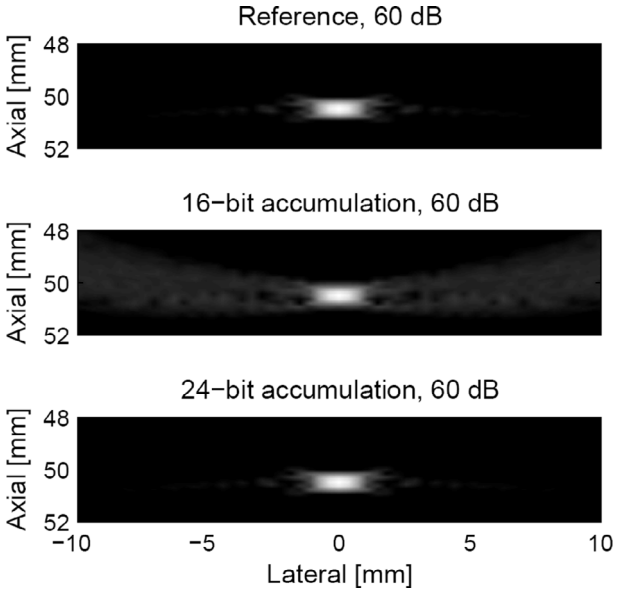


Fig. 10. A B-mode image of a point. The top plot shows a reference image and the bottom gives result of the beamformer using 16-bit numbers in the final summation procedure.

Fig. 11 shows several plots representing the results of several test cases. Table III summarizes which processing steps are present in each test case. All point spread functions have been calculated using exactly the same steps. Gradually the double-precision parameters are replaced by integer parameters as shown in the table. As it can be seen from Fig. 11, the results from all test cases, except for the 16-bit accumulation, are rather similar. The main lobes are virtually indistinguishable and the side-lobe levels are concentrated within 1 to 2 dB and are below −60 dB from the peak value.

The largest influence on the image is the noise introduced by the rounding operations. The results in the 16-bit accumulation represent the most conservative design. In this case, the side-lobe level is about 50 dB below the peak value of the point spread function. The energy in the side lobes is about 40 dB below the energy of the main lobe as shown in Fig. 11. Due to attenuation, signals from larger depths will rarely use the full 12-bit range of the A/D converter. The $4 \times \lambda$ sampling means that the 2 adjacent samples used by the linear interpolation will not be close to the maximum value simultaneously. Therefore, there is no need to downshift the result from the linear interpolation. Summing 24 bit across boards will, thus, not introduce any noise, and the resulting image will be comparable to the reference image.

## VI. Summary and Conclusion

This paper describes a beamformer implemented using modern programmable devices (FPGA). The beamformer consists of beamforming blocks that process the data in parallel. A single device can house 3 beamforming blocks and is capable of beamforming data from 4 transducer ele-

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.

TABLE III
A SUMMARY OF THE TEST CASES.

| Experiment | $4 \times \lambda$ Sampling | 12-bit quantization | Integer delays | Integer apodization | 24-bit accumulation | 16-bit accumulation |
|---|---|---|---|---|---|---|
| Reference | • | | | | | |
| PSF 1 | • | • | | | | |
| PSF 2 | • | • | • | | | |
| No rounding | • | • | • | • | | |
| 24-bit sum | • | • | • | • | • | |
| 16-bit sum | • | • | • | • | | • |

[1]Implemented using serial links.

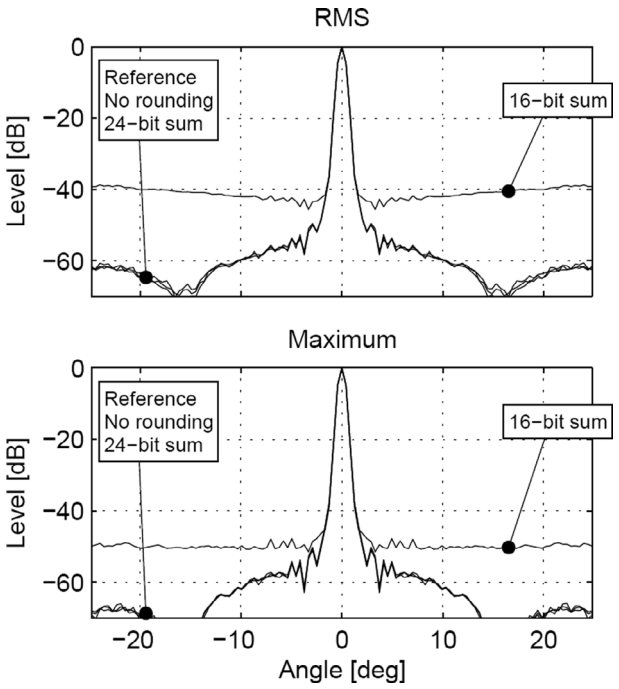


Fig. 11. Projection of the point spread function as a function of angle. The top plot shows the RMS and the bottom, the maximum value. The precision of the delays and apodization coefficients is sufficient to get a result that is less then 1 dB different than the reference. The major contributors to reduced image quality are the rounding operations in the beamforming process.

ments and producing 450 (150 per block) million complex (IQ) samples per second. This is equivalent to forming 4600 low-resolution images consisting of 96 lines, each long 1024 samples, because this number is sufficient to meet Nyquist's sampling criterion. If the lines are shorter, then their number can be increased. The suggested beamformer can handle images with arbitrary geometry, as long as they can be described as a collection of lines. Each line can have arbitrary origin, direction, and inter-sample distance. The length of the line is limited to 1024 samples. Another restriction of the beamformer is the size of the input buffer, which is 4096 complex samples (IQ).

To make the implementation of the beamformer possible, several design ideas have been employed. Because it is impossible to read all imaging parameters from an external memory for the desired image size, we have developed a recursive parametric delay calculation unit. It can calculate the delays with a precision of $\lambda/128$ without employing any multipliers, thus sparing resources for the interpolation unit. Similarly, the apodization coefficients are calculated using a piecewise linear approximation. The results from the individual transmissions are accumulated internally in the device, thus reducing the requirements for input/output bandwidth by the number of transmissions $M$. To increase the image size, we have used the fact that the memory bandwidth to read data are 8 times faster than the rate at which data are coming to the FPGA. The output of the FPGA is time-delayed by 1 frame. Only 1/4th of the image is beamformed using all transmission. Then the frame is sent further. Then another quarter of the image is processed, and so on.

The delay generation has been tested and shown to perform with the expected accuracy. Tests have shown that the nonperiodic nature of the error prevents the forming of secondary grating lobes. The use of integer delays and apodization coefficients does not worsen the image quality, which is mainly determined by the sampling frequency in combination with linear interpolation. The single largest source of reduction in image quality is the introduction of noise due to rounding operations in the data path. Experiments have, however, shown that it is not necessary to truncate the result, because the results are well within the range of numbers handled by the hardware. The point spread function of the images created using the described beamformer have the same main lobe as the reference images produced using double-precision floating point calculations for the delays, the apodization coefficients, and in the interpolation procedure. The side-lobe level is about 1 dB higher, but still below −60 dB when the suggested architecture is used in combination with 24-bit summation for the end result. When 16-bit summation is used, the side lobe is about 50 dB below the peak value.

The present work has shown that it is possible to create real-time synthetic transmit aperture system using present technology. The quality of the images is close to what is attainable using off-line processing with high precision. The described beamformer architecture is implemented in hardware and will be used further to assess the clinical relevance of the use of synthetic aperture focusing for medical applications.

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.

## Acknowledgments

The authors of the paper would like to thank Hans Holten-Lund, presently employed by Prevas A/S, Copenhagen, Denmark, for fruitful discussions regarding the implementation of the beamformer.

## References

[1] M. Soumekh, *Synthetic Aperture Radar. Signal Processing with MATLAB Algorithms*. New York: John Wiley & Sons, Inc., 1999.

[2] J. J. Flaherty, K. R. Erikson, and V. M. Lund, "Synthetic aperture ultrasound imaging systems," U.S. Patent 3 548 642, Dec. 22, 1967.

[3] C. B. Burckhardt, P.-A. Grandchamp, and H. Hoffmann, "An experimental 2 MHz synthetic aperture sonar system intended for medical use," *IEEE Trans. Sonics Ultrason.*, vol. 21, no. 1, pp. 1–6, Jan. 1974.

[4] P. D. Corl, P. M. Grant, and G. S. Kino, "A digital synthetic focus acoustic imaging system for NDE," in *Proc. IEEE Ultrason. Symp.*, 1978, pp. 263–268.

[5] G. S. Kino, D. Corl, S. Bennett, and K. Peterson, "Real time synthetic aperture imaging system," in *Proc. IEEE Ultrason. Symp.*, 1980, pp. 722–731.

[6] D. K. Peterson and G. S. Kino, "Real-time digital image reconstruction: A description of imaging hardware and an analysis of quantization errors," *IEEE Trans. Sonics Ultrason.*, vol. 31, pp. 337–351, 1984.

[7] K. Nagai, "A new synthetic-aperture focusing method for ultrasonic B-scan imaging by the Fourier transform," *IEEE Trans. Sonics Ultrason.*, vol. SU-32, no. 4, pp. 531–536, 1985.

[8] G. E. Trahey and L. F. Nock, "Synthetic receive aperture imaging with phase correction for motion and for tissue inhomogenities—Part II: Effects of and correction for motion," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 39, pp. 496–501, 1992.

[9] M. O'Donnell and L. J. Thomas, "Efficient synthetic aperture imaging from a circular aperture with possible application to catheter-based imaging," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 39, pp. 366–380, 1992.

[10] M. Karaman, P. C. Li, and M. O'Donnell, "Synthetic aperture imaging for small scale systems," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 42, pp. 429–442, 1995.

[11] M. Karaman and M. O'Donnell, "Subaperture processing for ultrasonic imaging," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 45, pp. 126–135, 1998.

[12] J. T. Ylitalo and H. Ermert, "Ultrasound synthetic aperture imaging: Monostatic approach," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 41, pp. 333–339, 1994.

[13] J. T. Ylitalo, "Synthetic aperture ultrasound imaging using a convex array," in *Proc. IEEE Ultrason. Symp.*, 1995, pp. 1337–1340.

[14] J. T. Ylitalo, "On the signal-to-noise ratio of a synthetic aperture ultrasound imaging method," *Eur. J. Ultrasound*, vol. 3, pp. 277–281, 1996.

[15] C. H. Frazier and W. D. O'Brien, "Synthetic aperture techniques with a virtual source element," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 45, pp. 196–207, 1998.

[16] M. H. Bae, M. K. Jeong, T. K. Song, and Y. B. Ahn, "Experimental study of transmit synthetic focusing combined with receive dynamic focusing in B-mode ultrasound imaging systems," in *Proc. IEEE Ultrason. Symp.*, 1999, pp. 1261–1264.

[17] S. I. Nikolov, K. Gammelmark, and J. A. Jensen, "Recursive ultrasound imaging," in *Proc. IEEE Ultrason. Symp.*, vol. 2, 1999, pp. 1621–1625.

[18] M. H. Bae and M. K. Jeong, "A study of synthetic-aperture imaging with virtual source elements in B-mode ultrasound imaging systems," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 47, pp. 1510–1519, 2000.

[19] S. I. Nikolov, "Synthetic aperture tissue and flow ultrasound imaging," Ph.D. dissertation, Ørsted•DTU, Tech. Univ. Denmark, Lyngby, Denmark, 2001.

[20] S. I. Nikolov and J. A. Jensen, "3D synthetic aperture imaging using a virtual source element in the elevation plane," in *Proc. IEEE Ultrason. Symp.*, vol. 2, 2000, pp. 1743–1747.

[21] S. I. Nikolov and J. A. Jensen, "Virtual ultrasound sources in high-resolution ultrasound imaging," in *Proc. SPIE—Progress in Biomedical Optics and Imaging*, vol. 3, 2002, pp. 395–405.

[22] Y. Takeuchi, "Chirped excitation for $<-100$ dB time sidelobe echo sounding," in *Proc. IEEE Ultrason. Symp.*, 1995, pp. 1309–1314.

[23] J. Shen and E. S. Ebbini, "A new coded-excitation ultrasound imaging system—Part 1: Basic principles," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 43, no. 1, pp. 131–140, 1996.

[24] T. X. Misaridis, K. Gammelmark, C. H. Jørgensen, N. Lindberg, A. H. Thomsen, M. H. Pedersen, and J. A. Jensen, "Potential of coded excitation in medical ultrasound imaging," *Ultrasonics*, vol. 38, pp. 183–189, 2000.

[25] R. Y. Chiao and L. J. Thomas, "Synthetic transmit aperture using orthogonal Golay coded excitation," in *Proc. IEEE Ultrason. Symp.*, 2000, pp. 1469–1472.

[26] S. I. Nikolov and J. A. Jensen, "Comparison between different encoding schemes for synthetic aperture imaging," in *Proc. SPIE—Progress in Biomedical Optics and Imaging*, vol. 3, 2002, pp. 1–12.

[27] C. R. Cooley and B. S. Robinson, "Synthetic aperture imaging using partial datasets," in *Proc. IEEE Ultrason. Symp.*, 1994, pp. 1539–1542.

[28] R. Y. Chiao, L. J. Thomas, and S. D. Silverstein, "Sparse array imaging with spatially-encoded transmits," in *Proc. IEEE Ultrason. Symp.*, 1997, pp. 1679–1682.

[29] G. R. Lockwood, J. R. Talman, and S. S. Brunke, "Real-time 3-D ultrasound imaging using sparse synthetic aperture beamforming," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 45, pp. 980–988, 1998.

[30] K. L. Gammelmark and J. A. Jensen, "Multielement synthetic transmit aperture imaging using temporal encoding," *IEEE Trans. Med. Imag.*, vol. 22, no. 4, pp. 552–563, 2003.

[31] M. H. Pedersen, K. L. Gammelmark, and J. A. Jensen, "In-vivo evaluation of convex array synthetic aperture imaging," *Ultrasound Med. Biol.*, vol. 33, pp. 37–47, 2007.

[32] J. S. Jeong, J. S. Hwang, M. H. Bae, and T. K. Song, "Effects and limitations of motion compensation in synthetic aperture techniques," in *Proc. IEEE Ultrason. Symp.*, 2000, pp. 1759–1762.

[33] H. S. Bilge, M. Karaman, and M. O'Donnell, "Motion estimation using common spatial frequencies in synthetic aperture imaging," in *Proc. IEEE Ultrason. Symp.*, 1996, pp. 1551–1554.

[34] S. I. Nikolov and J. A. Jensen, "In-vivo synthetic aperture flow imaging in medical ultrasound," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 50, pp. 848–856, 2003.

[35] J. A. Jensen and S. I. Nikolov, "Directional synthetic aperture flow imaging," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 51, pp. 1107–1118, 2004.

[36] C. R. Hazard and G. R. Lockwood, "Theoretical assessment of a synthetic aperture beamformer for real-time 3-D imaging," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 46, pp. 972–980, 1999.

[37] S. I. Nikolov, J. A. Jensen, and B. G. Tomov, "Processing in SARUS," Ørsted•DTU, Tech. Univ. Denmark, Tech. Rep., 2006.

[38] J. A. Jensen, M. Hansen, B. G. Tomov, S. I. Nikolov, and H. Holten-Lund, "System architecture of an experimental synthetic aperture real time ultrasound system," in *Proc. IEEE Ultrason. Symp.*, Oct. 2007, pp. 636–640.

[39] H. T. Feldkämper, R. Schwann, V. Gierenz, and T. G. Noll, "Low power delay calculation for digital beamforming in handheld ultrasound systems," in *Proc. IEEE Ultrason. Symp.*, vol. 2, 2000, pp. 1763–1766.

[40] B. G. Tomov and J. A. Jensen, "Delay generation methods with reduced memory requirements," in *Proc. SPIE—Medical Imaging*, 2003, pp. 491–500.

[41] J. A. Jensen and S. I. Nikolov, "A method for real-time three-dimensional vector velocity imaging," in *Proc. IEEE Ultrason. Symp.*, 2003, pp. 1582–1585.

[42] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C. The Art of Scientific Computing*. Cambridge: Cambridge University Press, 1988.

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.

[43] J. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electronic Computing*, vol. EC-3, no. 3, pp. 330–334, Sep. 1959.
[44] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. Spring Joint Computer Conf.*, 1971, pp. 379–385.
[45] S. I. Nikolov, J. A. Jensen, and B. G. Tomov, "Recursive delay calculation unit for parametric beamformer," in *Proc. SPIE—Progress in Biomedical Optics and Imaging*, vol. 6147-13, 2006, pp. 1–12.
[46] R. G. Pridham and R. A. Mucci, "Digital interpolation beamforming for low-pass and bandpass signals," *Proc. IEEE*, vol. 67, no. 6, pp. 904–919, June 1979.
[47] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay," *IEEE Sig. Proc. Mag.*, vol. 13, no. 1, pp. 30–60, 1996.
[48] Xilinx. (2007, Sept.). "Virtex-4 family overview (v3.0) (ds112.pdf)." Product Specification. [Online]. Available: http://www.xilinx.com.
[49] Xilinx, *DSP: designing for optimal results. High-performance DSP using Virtex-4 FPGAs*, Xilinx Inc., 2005.
[50] S. Holm and K. Kristoffersen, "Analysis of worst-case phase quantization sidelobes in focused beamforming," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 39, pp. 593–599, 1992.
[51] J. A. Jensen and N. B. Svendsen, "Calculation of pressure fields from arbitrarily shaped, apodized, and excited ultrasound transducers," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 39, pp. 262–267, 1992.
[52] J. A. Jensen, "Field: A program for simulating ultrasound systems," *Med. Biol. Eng. Comp.*.

**Jørgen Arendt Jensen** (M'93–SM'02) earned his Master of Science degree in electrical engineering in 1985 and his Ph.D. degree in 1989, both from the Technical University of Denmark. He received the Dr.Techn. degree from the same university in 1996.

He has published more than 160 journal and conference papers on signal processing and medical ultrasound and the book *Estimation of Blood Velocities Using Ultrasound*, published by Cambridge University Press in 1996. He is also developer of the Field II simulation program. He has been a visiting scientist at Duke University, Stanford University, and the University of Illinois at Urbana-Champaign. He is currently full professor of Biomedical Signal Processing at the Technical University of Denmark in the Department of Electrical Engineering and head of the Center for Fast Ultrasound Imaging and the Section for Biomedical Enginering. He is also adjunct full professor at the Faculty of Health Sciences at the University of Copenhagen. He has given courses on blood velocity estimation at both Duke University and University of Illinois and teaches biomedical signal processing and medical imaging at the Technical University of Denmark. He has given several short courses on simulation, synthetic aperture imaging, and flow estimation at international scientific conferences. He has received several awards for his research. He is also the co-organizer of a new biomedical engineering education offered by the Technical University of Denmark and the University of Copenhagen. His research is centered around simulation of ultrasound imaging, synthetic aperture imaging, vector blood flow estimation, and construction of ultrasound research systems.

**Svetoslav I. Nikolov** received the M.Sc. degree in electrical engineering and the M.B.A. degree in international business relations from the Technical University of Sofia in 1996 and 1997, respectively. In 2001, he received his Ph.D. degree from the Technical University of Denmark, Lyngby. His dissertation explored approaches for synthetic aperture tissue and flow imaging, and possibilities for real-time 3D imaging.

After completing his doctoral work, he stayed on at Ørsted•DTU, Technical University of Denmark as an assistant professor, where he taught digital design, software development, and digital signal processing. Presently Svetoslav Nikolov is an Associate Professor at the Department of Electrical Engineering. His current research interests are focused on real-time systems and signal processing, and methods for high-resolution tissue and flow imaging.

**Borislav Gueorguiev Tomov** was born on November 28, 1973, in Montana, Bulgaria. He earned the M.Sc. degree in electronics from the Technical University of Sofia, Bulgaria, in 1996, and the Ph.D. degree from the Danish Technical University, Denmark, in 2003. He is currently an Assistant Professor at the latter university. His research interests include ultrasound imaging and digital signal processing.

Authorized licensed use limited to: Danmarks Tekniske Informationscenter. Downloaded on November 4, 2009 at 10:12 from IEEE Xplore. Restrictions apply.